Moving Least-Squares Surfaces

Carlos Eduardo Scheidegger - cscheid@sci.utah.edu
Geometric Processing - CS 7960 - Prof. Cláudio Silva
Introduction

- Name of the game is scattered data approximation
Moving Least Squares in a nutshell

- Take the classic Least Squares problem
- Make the weights change “as you move” on the surface
Why not RBFs?

- RBFs beautifully solve the interpolation problem
- Given a set of points, find the “piano wire” solution: Peg all the points in 3D, find the surface that interpolates and “bends the least”
  \[ \int_{\Delta} f_{xx}^2 + f_{xy}^2 + f_{yy}^2 \, d\Delta \]
- In a few words: it’s easy and quick to make it work...
- ... If we don’t care about implementation - slow, ill-conditioned, global - large \( n \times n \) dense matrix
- Only interpolates (easily) - regularization is really a hack
Surfaces defined by Moving Least Squares ...
Overview

- Given some basis functions $b^{(i)}$ such that $b^{(0)} = 1$, $b^{(i)} \in C^m(D)$ independent over “some $S \subset D$”

- for each $\hat{z} \in \bar{D}$, we want to find local approximations $L_{\hat{z}} f$

- $L_{\hat{z}} f = \sum_i a_i(\hat{z})b^{(i)}$

- Then, for any $z$,

  $G f(\hat{z}) = L_{\hat{z}} f(\hat{z}) = \sum_i a_i(\hat{z})b^{(i)}(\hat{z})$

- $G f$ is simply the local approximation at the given point
Inner Product Spaces

- Vector space $(V, +, \cdot)$ equipped with $\langle \cdot, \cdot \rangle$

- $\langle a, b \rangle = \langle b, a \rangle$

- $\langle a, a \rangle \geq 0, \langle a, a \rangle = 0$ iff $a = 0$

- $\langle s \cdot a, b \rangle = s \cdot \langle a, b \rangle$
Inner Product Spaces

- These need not be vectors!
- Continuous functions, for example
  - Many different inner products:
    - Continuous $L_2$: $\langle f, g \rangle = \int_a^b f(x)g(x) \, dx$
    - Continuous $L_\infty$: $\langle f, g \rangle = \max |f(x) - g(x)|, a \leq x \leq b$
  - The one we’ll use: Discrete, $\hat{z}$-dependent $L_2$: $\langle f, g \rangle = f^T W(\hat{z}) g$
Inner Product Spaces

- $\langle f, g \rangle = f^T W(\hat{z}) g$
- $f = (f(z_1), f(z_2), f(z_3), \cdots, f(z_N))^T$
- $\langle f, g \rangle = \sum_i f(z_i) g(z_i) w^{(i)}(\hat{z})$
- Our inner product space is just a weighted least squares, whose weights are dependent on the point in space
- The weights move
Approximation Theory in two minutes

Quick recap for regular least squares:  \( \min \|Ax - b\|^2 \)

\[
\nabla \langle Ax - b, Ax - b \rangle = 0
\]
\[
\nabla (Ax - b)^T (Ax - b) = 0
\]
\[
2A^T (Ax - b) = 0
\]
\[
A^T (b - Ax) = 0
\]
\[
A^T Ax = A^T b
\]

The Normal Equations
Approximation Theory in two minutes

- Given an inner product space, a function space $F$ and a set of basis functions that space $\tilde{F} = \text{span}\{b^{(i)}\} \subset F$

- A least squares approximation satisfies

$$\langle f - L\hat{z} f, b^{(i)} \rangle = 0$$

- The error is \textbf{orthogonal} to $\tilde{F}$
The Moving Least Squares case...

We can expand $L_{\hat{z}} f$ from the definition:

$$L_{\hat{z}} f = \sum_{i} a_i(\hat{z}) b^{(i)}$$

$$\langle f - L_{\hat{z}} f, b^{(i)} \rangle = 0$$

$$\sum_{i} a_j(\hat{z}) \langle b^{(i)}, b^{(j)} \rangle = \langle f, b^{(i)} \rangle$$
... is just like the original!

\[ \sum_i a_j(\hat{z}) \langle b^{(i)}, b^{(j)} \rangle = \langle f, b^{(i)} \rangle \]

\[ (b^{(j)})^T = (b^{(j)}(z_1), \ldots, b^{(j)}(z_N)) \]

\[ BW(\hat{z}) B^T a = BW(\hat{z}) f \quad \quad A^T Ax = A^T b \]

- When \( b^{(i)} \) are monomials, \( B \) is the Vandermonde matrix
Is this still well-defined?

- Is there a single minimum?
- Equivalently, is $BWB^T$ symmetric positive-definite?
Is this still well-defined?

- We know $W$ is symmetric positive-definite
- Remember that $A$ is SPD iff $x^T A x > 0, x \neq 0$
- We simply say $y^T = x^T B$
- Follows directly

\[ x^T A x > 0, x \neq 0 \]
This is where we stop

- A lot more (interesting, unimportant) math for interpolating MLS schemes
- Idea: make weight go to infinity as distance goes to 0
Mesh-independent Surface Interpolation

Before
- Input: points, domain
- Output: function in the same* domain

Now
- Input: points, domain of embedding space, dimension of desired output
- Output: manifold
Challenge

- Manifold: space that’s “locally like a plane”
- “Locally like a plane” equivalent to “a function over some reference frame”
- The big problem is to find the reference frame
MLS Surfaces
Starting with what doesn’t work

- Let’s take the previous technique and adapt it directly
- Levin claims it’s not a projection
- Minimize $\sum_i (p(x_i) - f_i)^2 \theta(||r_i - r||)$
- First, what does that equation mean?
Notation changes

\[ \sum_i (p(x_i) - f_i)^2 \theta(||r_i - r||) \]

- \( r_i \): input points
- \( H \): plane in consideration
- \( x_i \): projections of input points
- \( f_i \): height of \( r \) over \( H \)
- \( r \): point to be projected

H shows up nowhere, but most of the definition depends on it. That’s ugly!
Why does it not work?

- The operator $P$ is not a projection
- In other words, $P^2(r) \neq P(r)$
The corrected definition

- find $H$ that minimizes
  \[ \sum_i (\langle a, r_i \rangle - D)^2 \theta(||r_i - q||) \]

- $a$ is the normal of $H$
- $q$ is the center of the reference frame (any point of $H$)
- Big difference: weights are defined by distance to center of reference frame
  - Weights won’t change along normal of $H$
  - Weights change with $q$: non-linear
Some conditions

- $q$ must minimize $J$
- $a$ must be parallel to $q - r$
- The directional derivative of the error, in the direction of $a$, must be zero

$$\frac{\partial J(q)}{\partial \vec{a}} = 0$$
Why is it a projection?

- (Note: I don’t like the author’s proof)
- Express everything in polar coordinates \((\rho, \theta_0, \cdots, \theta_{n-1})\) around \(r\)
- First constraint forces \(q = (||r - q||, 0, \cdots, 0)\)
- Second constraint says \(\frac{\partial J(q)}{\partial \rho} = 0\)
The Jacobian

- The Jacobian of an operator $O$ is the matrix of first partial derivatives.
- Let’s do examples with linear transformations: matrices.
  - $O(\vec{x}) = A\vec{x}$
  - $J(O) = \nabla(A\vec{x}) = A$
The Jacobian

It’s a projection - Jacobian is rank deficient!

\[
\begin{bmatrix}
\frac{3}{4} & \frac{\sqrt{3}}{4} \\
\frac{\sqrt{3}}{4} & \frac{1}{4}
\end{bmatrix} = \begin{bmatrix}
\frac{\sqrt{3}}{2} \\
\frac{1}{2}
\end{bmatrix} \cdot \begin{bmatrix}
\sqrt{3}/2 & 1/2
\end{bmatrix}
\]
Why is it a projection?

- Can we say something about the Jacobian of $P$?
- Again, polar coordinates
- How do $\rho, \theta_i$ change as we change $\rho$ of input?

$$\frac{\partial \rho_q}{\partial \rho_r} = \frac{\partial \theta_q}{\partial \rho_r} = 0$$

- Entire row of zeros!
Now for the easy part

- After the reference frame has been found, we look for a local best-approximating function

- Just like Lancaster and Salkauskas
Interpolation

Weirdness: $\theta(||q_i - q||)$
Point-Set Surfaces
Applications

- So we have this really neat mathematical result
- What do we do with it?
  - Upsampling
  - Downsampling
  - Rendering
- Computational Aspects (important!)
Review of the projection

- q is the center of the reference frame, H
- p_i are the input points
- f_i are the error of the points on H
- n is defined by r-q
- r is the point to be projected
- g is the local function over H
Review of the projection

- What happens to the error as \( q \) moves away from the points?
- (oops)
- Solution: partition of unity

\[
\sum_i (\langle n, p_i \rangle - D)^2 \theta(||p_i - q||) \\
\sum_i \theta(||p_i - q||)
\]
The weighting function

- Should be as smooth as you want your surface to be

\[ \theta(d) = e^{\frac{d^2}{h^2}} \]

- Update: no more fixed $h$
  - Compute them based on some local descriptor
Global support

- In principle, all points contribute to all of the surface
- Proposed idea: fast multipole-like clustering
- (What we do in practice): k nearest neighbors
Downsampling

- Iteratively remove “redundant” points
- Points that are close to the surface
- I think this favors keeping noisy points
Upsampling

- Compute local parameterization of surface
- Use polynomial from projection
- Compute Voronoi diagram on parameterization
- Upsample at Voronoi vertex with largest empty ball
Rendering

- We want to show a continuous surface all the time
- Upsampling might be necessary
- Reprojecting everything in real time is too slow
- Use polynomials
Rendering

- We want overlapping regions
- Not too much, though
- Define polynomial patch sizes carefully
Results
To be continued

- Next class
- Implicit surfaces will play a big role
  - Amenta’s paper shows a simple version of the original MLS is, in a sense, an implicit surface
  - Alexa’s linear versions of the MLS are all implicit surfaces
- Trouble with MLS!
  - The domain of a MLS surface